Introduction

- Motivation for Xen
- Difference of full virtualization and paravirtualization
- Experimental Results
- Updated Experimental Results
- Discussion
Motivation

- Computers are powerful
- VMs provide convenience
- VMs provide isolation
- VMs are cost efficient
Full Virtualization

- Provide ability to run an unmodified Guest
- Have to emulate whole world for Guest
Full Virtualization

- Processor must meet three requirements (Robin 2000)
  - Non-privileged instructions
  - Isolation
  - Automatic signaling
Non-Privileged Instructions

- Executing a non-privileged instruction must be roughly equivalent in privileged mode and user mode
- Processor could not use an additional bit or portion in privileged mode
Isolation

• Real system and all other VMs must be protected from errant VMs

• If one VM is getting a DOS attack it should not take anything else down (ISPs)

• Memory should be isolated
Isolation
Isolation

VM1
VMM1

VM2
VMM2

VM3
VMM3

Not Allowed

Memory1 Memory2 Memory3

Allowed
Automatic Signals

- Signal VMM when VM attempts to execute sensitive instructions
- Change or reference VM mode or state of machine
- Read or change sensitive registers or memory
- Reference memory outside of it’s virtual memory
Automatic Signals

Machine State | Sensitive Register | Clock Register | Interrupt Registers | Memory outside Virtual Memory

VM

VMM
Full Virtualization

• However, x86 did not support full virtualization
Privileged Instructions

x86

• Supported
• Privileged instructions and non-privileged instructions execute the same
• Privileged instructions cause exception if CPL is not 0
Isolation

x86

- Supported
- Provide protection mechanism
  - Descriptor Privilege Level 0-3 (Rings)
  - Enables boundaries to be enforced
Automatic Signals
x86

• Does provide traps and interrupts to redirect program execution
Automatic Signals

x86

• However, x86 does not satisfy requirements for all instructions
  • 17 out of 250
  • Instructions often give information about host, not guest
Automatic Signals x86
MOVE Instruction

- Stores CPL in bits 0 and 1
- Violates Non-Privileged vs Privileged
- Causes exception when task loads registers, privilege check
Virtualization

• Robin concluded that Intel Pentium was not appropriate for secure full virtualization in 2000
Full virtualization

• For full virtualization to work:
  • Must insert traps
  • Memory Translation
  • Security and Isolation
Cost?

- Large overhead for instruction translation
- Much of it up front
- Overhead for memory translations
Instructions

• Entire code must be ripped apart to scan for instructions and insert more instructions to go back into hypervisor

• All done in binary translation

• Hypervisor can only execute small chunks
Instructions
Memory

• Shadow Page Tables
  • VMware keeps Shadow
    • Logical
    • Physical
  • Hardware
Shadow Page Tables

- Figure 8.4 G. Back 07
Xen is different

- Run a modified guest with small adaptations
- Use the strengths of x86
- Provide mechanisms to support weaknesses
- Paravirtualization
Management

Control Plane Software

GuestOS (XenoLinux)
Xeno-Aware Device Drivers

GuestOS (XenoLinux)
Xeno-Aware Device Drivers

GuestOS (XenoBSD)
Xeno-Aware Device Drivers

GuestOS (XenoXP)
Xeno-Aware Device Drivers

Domain0 control interface
virtual x86 CPU
virtual phy mem
virtual network
virtual blockdev

H/W (SMP x86, phy mem, enet, SCSI/IDE)
Domain0

- Instantiates new Domains
- Allocates memory
- Allocates resources and privileges
- Management software
Memory Translation

- Guest OSes register directly with memory-management unit
- Have read only access to page tables
- Hypervisor must validate changes to page tables
Memory Translation

- Figure 8.4 G. Back 07
Memory Translation

- Uses Page frames to determine actions (PD, PT, LDT, GDT, RW)
- These are mutually exclusive
- Reference count must be 0 to change
CPU

• Use Rings for Privilege separation
  • 0(highest)-3(lowest)
  • Xen modifies the guest to run in Ring 1
• System calls and page faults are most frequent
  • System calls are sped up by ‘fast’ exception handler
  • Page faults are not able to use ‘fast’ exception handler (register CR2)
CPU Scheduling

- BVT no longer used
- Credit Based Scheduler
  - Fair
- Weight and Cap
- Schedules base on over and under credit
Paravirtualization: I/O

- No emulation
- Exposes simple abstractions
- Event used for asynchronous notification for domains
- However, now Xen leverages Existing Device Drivers
Paravirtualization: I/O

- Device must be separated from guest
- I/O Rings are used for each Domain
  - Both Hypervisor and Guest have access
I/O Ring

Request Consumer
Private pointer in Xen

Request Producer
Shared pointer updated by guest OS

Response Producer
Shared pointer updated by Xen

Response Consumer
Private pointer in guest OS

- **Request queue** - Descriptors queued by the VM but not yet accepted by Xen
- **Outstanding descriptors** - Descriptor slots awaiting a response from Xen
- **Response queue** - Descriptors returned by Xen in response to serviced requests
- **Unused descriptors**
Paravirtualization’s Protection

- Memory updates are validated by hypervisor
- Guest code runs in Ring 1
- Hypervisor terminates guest OS when a double fault is detected
- Hypervisor monitors I/O, buffers must be inside domain’s memory reservation
More Efficient?

- Gets rid of inserting traps by using Rings
- No shadow tables to keep
  - Avoids TLB flushes
- I/O resource visibility
Limitations

• Has to have a modified guest to run
• This has a high cost
• In fact at the time of this paper Xen ran on defined set of hardware
Experimental Setup

- Used RedHat Linux 7.2, kernel 2.4.21
- Measured Performance or
  - Macro
  - Micro
  - Concurrency
  - Isolation
Figure 3: Relative performance of native Linux (L), XenoLinux (X), VMware workstation 3.2 (V) and User-Mode Linux (U).
Concurrency

Figure 4: SPEC WEB99 for 1, 2, 4, 8 and 16 concurrent Apache servers: higher values are better.

Figure 5: Performance of multiple instances of PostgreSQL running OSDB in separate Xen domains. 8(diff) bars show performance variation with different scheduler weights.
Summary of Results

- Xen shines in high concurrency
- Significant step in performance
- Stated in paper that they significantly outperformed VMware ESX
State of Xen

• Xen supports many OSes
  • XP ran without modification
• Commercial application
• Competes with VMWare ESX
Updated Results

• VMware released updated results that showed it was more performant

• Xen’s response was that they incorrectly set up the experiments

• However...
As reported by Xen...

Figure 3 – SPECcpu INT 2000 results compared to native (higher values are better)
Performance

Figure 4 – Passmark – CPU results compared to native (higher values are better)
Performance

Figure 6 – Compile workload result compared to native (higher values are better)
Figure 9 – SPECjbb2005 results compared to native (higher values are better)
Performance

![Graph showing SPECjbb2005 results compared to native (higher values are better)]

Figure 11 – SPECjbb2005 results compared to native (higher values are better)
Performance

• These results came from Xen’s Whitepaper “A Performance Comparison of Commercial Hypervisors”

• Available on Xen’s website
Performance Results

• Xen has changed position from:
  • “we significantly outperform”

• To:
  • “we compete”
Xen
Expectations vs Reality

• Would outperform most tests
  • Seems to perform best on Linux with high-workloads
• Modifications would be small
  • Devices take a lot of code
VMware Performance

• Tests were performed on Windows Server 2003

• VMware requires less modifications

• VMware speculated at maturity and complexity problems
Discussion

• Was Xen correct in any assumptions it made?
• Market value of off the shelf support?
• VMware success proof of superiority...or just more resources?